Rossella Minoli

Graziano Pravadelli e Cristian Turetta

Reti di sensori e dispositivi indossabili

Progetto di reti di sensori e dispositivi indossabili

INTRODUZIONE

Questo progetto di reti di sensori e dispositivi indossabili mira a sviluppare un'applicazione che riconosce due azioni, nel mio caso, andare in bicicletta e saltare la corda. L'obiettivo è allenare la macchina in modo che essa riconosca le due azioni commettendo il minimo errore. Per sviluppare questa applicazione viene usato un codice python e il Thingy52. Quest'ultimo è un dispositivo sviluppato da Nordic Semiconductor e utilizzato per le applicazioni IoT(Internet of Things). Esso contiene diversi sensori ma per il progetto sono stati utilizzati l'accelerometro e il giroscopio. L'accelerometro è un sensore che misura l'accelerazione lineare del corpo su cui viene posizionato. Il principio che permette il funzionamento degli accelerometri, si basa sul fatto che una massa accelerata presenta una propria inerzia che può essere misurata. In questo caso l'accelerometro è di tipo triassiale cioè misura simultaneamente le variazioni di velocità nelle tre direzioni ortogonali (x, y e z). Il giroscopio, invece, è un sensore che misura la velocità angolare del corpo su cui è posizionato e permette di determinarne l'orientamento. Anch'esso è di tipo triassiale.

Il Thingy52 si connette ad altri dispositivi tramite il protocollo BLE (Bluetooth Low Energy) e consente la raccolta dati in tempo reale.

METODOLOGIA

Il codice python utilizzato per creare questa applicazione segue dei passaggi ben definiti.

Per la raccolta dati delle due azioni scelte, utilizziamo il nostro sensore, esso raccoglie dati sul movimento del corpo su cui è posizionato utilizzando l'accelerometro e il giroscopio. Per avviare questo processo, eseguiamo il file "main_collect_data". In questo file, inizialmente andiamo a specificare l'indirizzo del nostro sensore. Successivamente, tramite la funzione scan, cerchiamo tutti i dispositivi bluetooth vicini e, utilizzando la funzione find, identifichiamo l'indirizzo del nostro sensore. A questo punto, richiamando la classe Thingy52Client ci connettiamo al sensore e registriamo i dati, salvandoli in un file denominato "<codice del sensore>_<azione che stiamo registrando>.csv". La classe Thingy52Client contiene sostanzialmente tre parti. La prima parte è quella relazionata al sensore cioè di ricezione dei dati inerziali. La seconda parte si riferisce al salvataggio dei dati e la terza implementa la fase di inferenza, cioè calcola un modello per inferire i dati.

Dopo la raccolta dei dati, dobbiamo procedere alla fase di allenamento della macchina affinché possa riconoscere e distinguere le due azioni. Per questo, implementiamo una classe CNN, che definisce la creazione del nostro modello, e una classe CSVDataModule, progettata per gestire il caricamento e la preparazione dei dati per il nostro progetto di machine learning. Per l'addestramento dell'applicazione, eseguiamo il file "train", dove vengono definiti tutti i parametri necessari, compreso il numero di fold che mi servono per la validazione incrociata. Se, per esempio, scelgo un numero di fold pari a tre, nel dataset i dati vengono divisi in tre parti uguali e attraverso un ciclo for viene fatto l'addestramento su due slot di dati e viene testato il terzo. Alla fine del ciclo tutte e tre gli slot sono stati testati e la macchina è in grado di scegliere

il modello migliore, cioè quello che funziona meglio. Nel file “train”, viene eseguito un ciclo for in cui, per ogni fold, viene istanziato un modello. Successivamente viene salvato il modello migliore, cioè quello con una perdita di dati inferiore. Dopo queste operazioni, il modello scelto viene allenato e testato sui dati di valutazione.

Dopo l’esecuzione del file “train” si creano spontaneamente le matrici di confusione, in cui vengono graficate le volte in cui la macchina ha predetto in modo corretto l’azione in relazione alle volte in cui invece ha sbagliato. E’ una matrice nxn, dove n è il numero di azioni. Il risultato ottimale si ottiene quando la diagonale principale ha valori prossimi al 100 mentre in tutte le altre parti dovrebbero esserci valori prossimi allo zero.

Dopo l’operazione di addestramento della macchina rimane solo l’ultimo passaggio da implementare, testare, in tempo reale, se la macchina riconosce o meno le azioni. Per questo abbiamo creato il file “main_test”, questo ha la stessa struttura del file “main_collect_data” ma al posto di salvare i dati in un file, stampa a video la predizione in tempo reale. In questo modo, mentre qualcuno che indossa il sensore sta facendo dei movimenti si riesce a monitorare, attraverso lo schermo del computer, se la macchina riconosce o meno l’azione che viene eseguita.

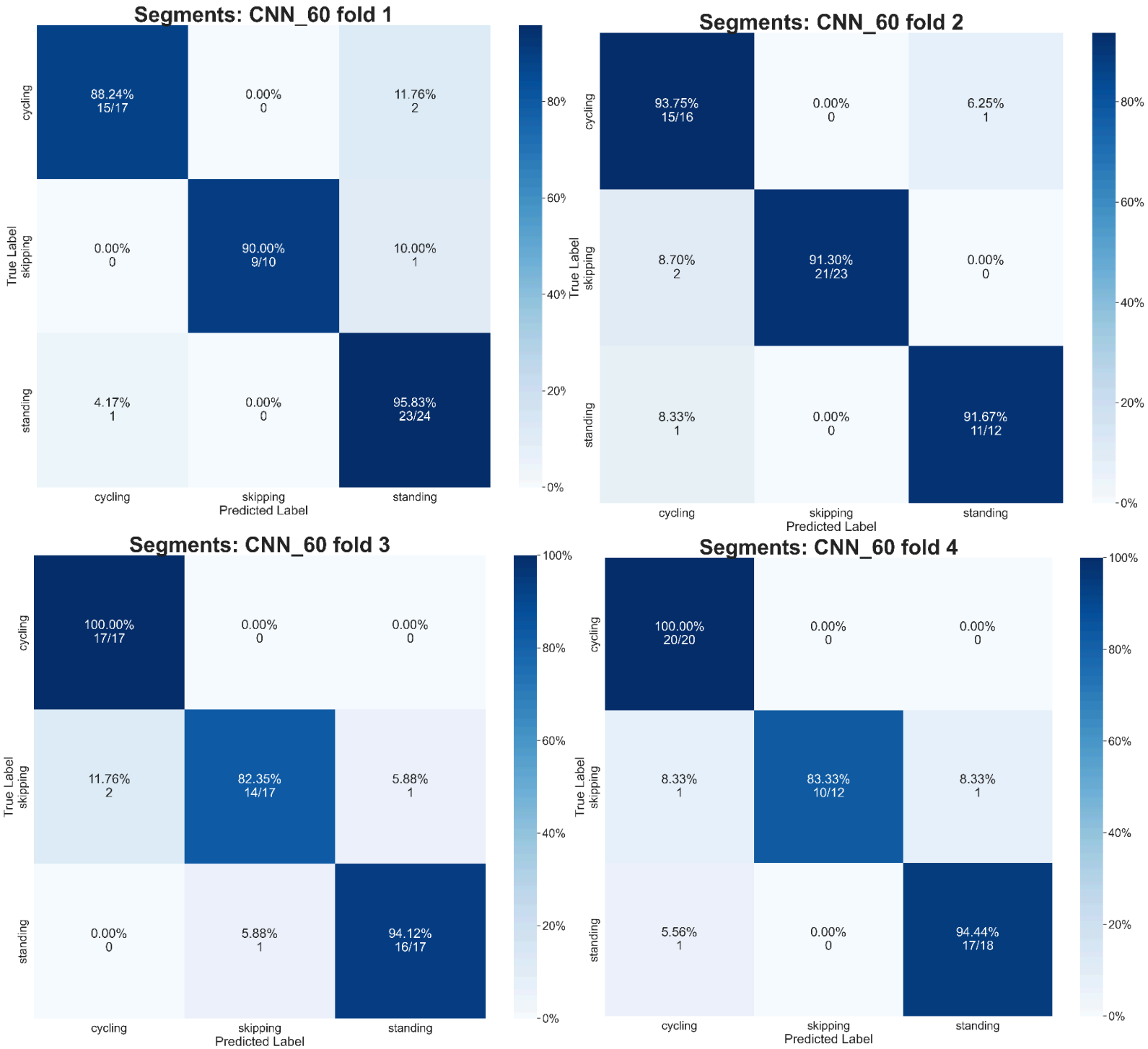
PARAMETRI SCELTI

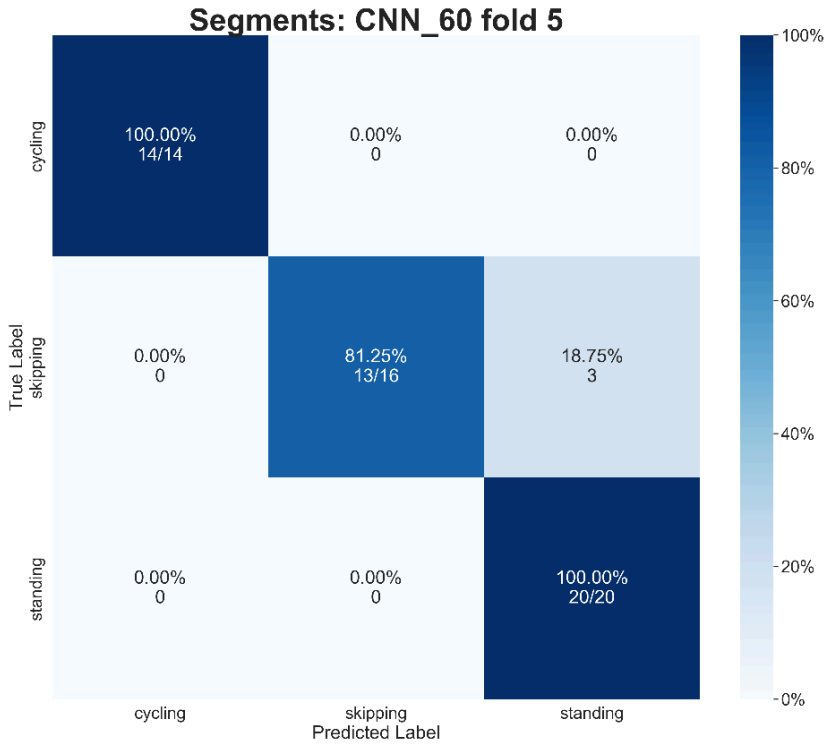
Per migliorare le matrici di confusione ottenute si possono modificare alcuni parametri definiti nel file “train”. Nel mio caso ho deciso di utilizzare questi parametri:

- k = 5, k sono i numeri di fold per la validazione incrociata, nel mio caso ho scelto il valore 5 poiché le matrici di confusione così risultano più coerenti ed esatte.

- window_size = 1, questo parametro descrive la dimensione della finestra temporale(in secondi) utilizzata per il preprocessing dei dati.
- sampling_frequency = 60, frequenza di campionamento (Hz) cioè la frequenza di registrazione dei dati.
- batch_size = 32, numero di campioni elaborati nella fase di train.

Matrici di confusione:





RISULTATI ATTESI

Nel mio caso, ho tentato di far riconoscere alla macchina due azioni specifiche: andare in bicicletta e saltare la corda. Ho raccolto i dati relativi a queste attività utilizzando le funzioni descritte nella metodologia, e successivamente ho addestrato il modello. Tuttavia, durante la fase di inferenza, ho osservato che sarebbe stato più opportuno inserire una terza azione, ovvero stare fermi. Mi aspetto che, includendo questa azione, la macchina sia in grado di distinguere meglio le due azioni principali senza commettere errori quando l'attività si interrompe.

CONCLUSIONE

In conclusione, questo progetto aveva l'obiettivo di far riconoscere alla macchina tre azioni specifiche, andare in bicicletta, saltare la corda e rimanere fermi in piedi, con un buon livello di

accuratezza. I risultati attesi però non sono stati raggiunti. Quando ho aggiunto la terza azione le matrici di confusione risultavano abbastanza corrette. Dopo aver confrontato i risultati con il numero di k-fold uguale a 3 e a 5, ho deciso di optare per il secondo caso perché le matrici risultanti erano più coerenti e adeguate. Tuttavia, quando ho testato il programma su altre persone, la predizione si è rilevata inaccurata. La macchina non ha compreso la differenza tra le tre azioni, nonostante i tentativi di modificare il numero di k-fold da 3 a 5 e di raccogliere nuovamente i dati. In questo caso potrebbe essersi verificato il problema dell'overfitting o sovradattamento, cioè il modello risulta accurato solo con gli esempi utilizzati in fase di training, mentre le sue performance degradano nella fase di test o in applicazioni successive. In altre parole, in questo caso il modello si è adattato eccessivamente ai dati inseriti e non è quindi in grado di generalizzare.